ARL-MR-0901 • AUG 2015

US Army Research Laboratory

Design Report for the Synchronized Position, Velocity, and Time Code Generator

by Brian T Mays

Approved for public release; distribution unlimited.

NOTICES

Disclaimers

ARL-MR-0901 ● AUG 2015

US Army Research Laboratory

Design Report for the Synchronized Position, Velocity, and Time Code Generator

by Brian T Mays
Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED (From - To)**
Aug 2015	Final	04/2015–06/2015

4. TITLE AND SUBTITLE	
Design Report for the Synchronized Position, Velocity, and Time Code Generator	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	**5d. PROJECT NUMBER**
Brian T Mays	R.0015868.1
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**
US Army Research Laboratory ATTN: RDRL-SES-S 2800 Powder Mill Road Adelphi, MD 20783-1138	ARL-MR-0901
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Intelligence Surveillance and Reconnaissance (ISR) Technology Integration Branch of the US Army Research Laboratory (ARL) developed a Synchronized Position, Velocity, and Time Code Generator (SPVTCG). The SPVTCG encodes time, position, and velocity information derived from the global positioning system (GPS) and produces synchronized digital waveforms aligned with the GPS pulse per second (PPS) events. The alignment of the data with the PPS provides the synchronization between remote sites and serves as the reference channel in a distributed data acquisition system.

15. SUBJECT TERMS

GPS Time Code Generator, IRIG, Data Synchronization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Brian T Mays
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	24	301-394-2185

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

Mark Ware of the US Army Research Laboratory (ARL) is acknowledged for his contribution of the printed circuit board (PCB) layout of the global positioning system (GPS) Time Code Generator.

Brian Mary of ARL is acknowledged for his contribution of the enclosure design of the GPS Time Code Generator.

INTENTIONALLY LEFT BLANK.

1. Introduction

1.1 Background

The Intelligence Surveillance and Reconnaissance (ISR) Technology Integration Branch of the US Army Research Laboratory (ARL) developed a Synchronized Position, Velocity, and Time Code Generator (SPVTCG), shown in Fig. 1. The SPVTCG encodes time, position, and velocity information derived from the global positioning system (GPS) and produces synchronized digital waveforms aligned with the GPS pulse per second (PPS) events. The alignment of the data with the PPS provides the synchronization between remote sites and serves as the reference channel in a distributed data acquisition system. The SPVTCG outputs are designed to be directly captured in parallel with raw sensor data as digital or analog input channels. This tight coupling with the sensor data streams insures precise position, velocity, and time information is recorded across multiple collection sites.

Fig. 1 Top view of the unit with the lid removed

The motivation to develop the SPVTCG was to address many of the shortcomings of traditional methods such as recoding Inter-Range Instrumentation Group (IRIG) time codes only in the data streams. The IRIG standards date back to the 1960s and have provided a robust means for distributed time synchronization to support data collection. One of the key limitation with IRIG is that it only encodes time information onto the reference channel. With the establishment of GPS, not only is precise time information available but also position and velocity. All 3 parameters are critical to a nonstationary distributed data collection system. Even in the case of stationary systems, position information directly embedded in the data streams reduce ambiguity in the data sets and improves integrity. Common solutions are to record IRIG signals as 1 of the sensor data channels and collect the GPS National Marine Electronics Association (NMEA) messages on an independent serial port for later processing and association to derive position, velocity, and time (PVT) information. This approach can prove problematic when data sets are subdivided in to very short segments for post-processing. While NMEA messages can be recorded in parallel with the IRIG channel, this has the drawback of requiring a sample rate sufficient to preserve the NMEA data, which can be much higher than the sample rate required of the primary sensor data being recorded. For these reasons, the SPVTCG was developed to provide a low data rate reference channel to be used in support of distributed data collection.

1.2 Theory of Operation

The SPVTCG derives all of its information from a low-cost GPS receiver. The specific receiver used in this design is the NEO-6 manufactured by u-blox. The selection of the GPS receiver is driven by size and performance, as all standard GPS receivers will produce the equivalent information needed by the SPVTCG. The information required is a PPS signal and a data channel containing the PVT information. The PVT information can be derived from standard NMEA messages or proprietary binary messages for efficiency. The PVT data are packetized and transmitted in time synchronization with the PPS events. To reduce the bandwidth requirements of the data capture system, PVT data are sent as individual packets of data once per second. This produces on report of each type every 3 s in the data stream; since all packets are sent synchronized to PPS events, timestamps are still maintained at a 1-Hz rate.

The SPVTCG produces 3 types of data packets, which are all 20 bytes in length and sent at a rate of 200 bits/s. This produces a transmission window of 800 ms with a 200-ms blank period prior to the next PPS event. This was done to allow easy framing of the data and provide a clear indication of the PPS event relative to the sample clock in the recorded data stream. To aid in the data demodulation, a second digital clock channel is provided to recover the data directly. The general intention is that both channels be recoded on a digital input channel of the data acquisition system captured synchronously with the raw sensor data. If digital channels are not available, SPVTCG data output can be captured on single analog channel and the data recovered, since all packets start with a fixed bit pattern of 0xA5. This allows clock recovery and subsequent data decoding if single channel data recording is used.

A block diagram of the SPVTCG in shown in Fig. 2. The primary components are the GPS receiver made by u-blox, a GPS patch antenna by Taoglas, an 8-bit microprocessor by Microchip, and a DC/DC converter design based on Simple Switcher by Texas Instruments. The unit accepts DC power in the range of 8–50 V and has 2 digital outputs that are 5-V transistor-transistor logic (TTL) compatible. The schematics of the SPVTCG are provided in Appendix A for complete reference.

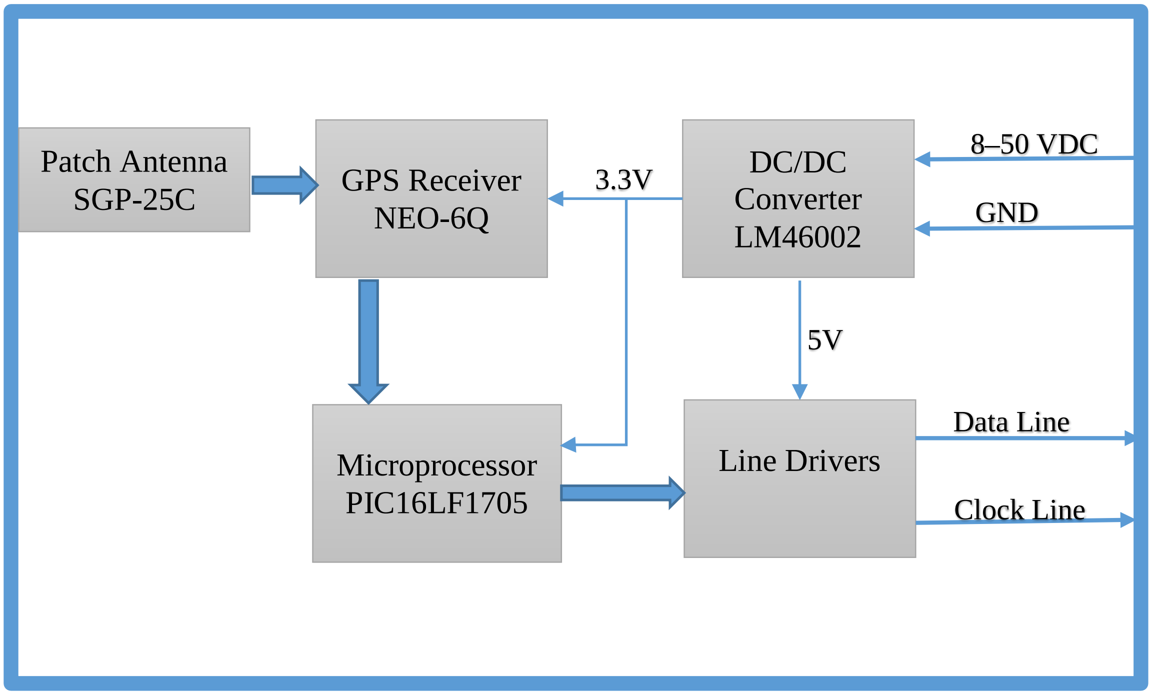

Fig. 2 SPVTCG block diagram

2. Interface Control Documentation

2.1 Electrical Interface

All electrical inputs and outputs are brought out on a single 4-pin military connector.

The connector specification and pin assignments are as follows:

Connector type: Amphenol PT02E-8-4P

Pin-Outs:
- A - VBAT+ (8–50 VDC)
- B - GND (power and signal)
- C – SPVTCG data (5 V TTL)
- D – SPVTCG clock (5 V TTL)

Power Consumption (typ. at 25 °C): 0.5 W

2.2 Binary Data Stream Specification

A sample binary data stream captured by a logic analyzer, shown in Fig. 3, demonstrates the basic structure of all the data packets transmitted by the SPVTCG. Note that channels 3 and 4 in the figure represent the SPVTCG clock and data, respectively. The signal on channel 1 is the internal PPS signal directly from the GPS receiver and is not available on the external connector. The PPS signal is important here to show the relationship that the leading edge of both the SPVTCG clock and SPVTCG data align with PPS and provide the timing information. The SPVTCG clock is only active during valid data transmission and therefore has an envelope of 800 ms. The SPVTCG data packets also have the same envelope and all start with 0xA5 guaranteeing a leading positive edge aligned to PPS on that channel. The packetized data are sent least significant bit first and the data should be read on the falling edge of SPVTCG clock.

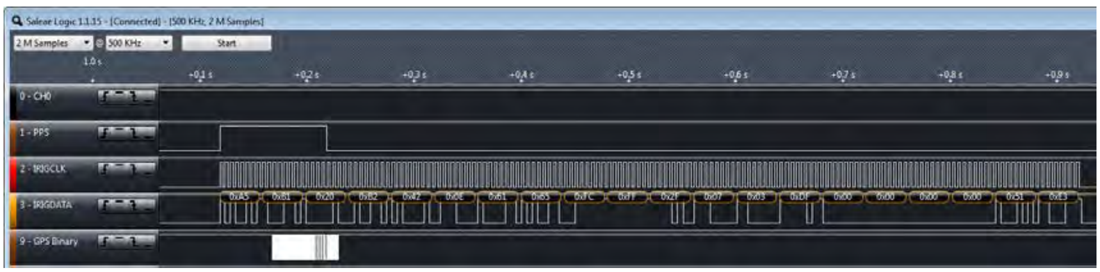

Fig. 3 Logic analyzer capture of raw data streams

2.3 Data Packet Format Specification

The data packets are 20 bytes long with the following format (Table 1):

- A 1-byte start of packet (SOP), defined as 0xA5
- A 1-byte message identification (MID)
- A 16-byte message payload, all multiple byte data are transmitted as little-endian, integer values
- A 2-byte checksum produced with the 8-bit Fletcher algorithm as specified by RFC 1145 and is calculated over the entire packet: SOP, MID, and payload.

Table 1 Data packet format structure

SOP	MID	Payload	Checksum
0xA5	See below	See below	CHK[0] CHK[1]

Example: 0xA5B120B2420E6165FCFF2F0703DF0000000051E3

2.3.1 Individual Message Definition

Tables 2 through 5 show the individual MID payload specifications: PPS time, position, velocity, and no solution message. Note in the format columns in the tables that (U) indicates unsigned integer values and (I) indicates signed integer values. The numbers in the format columns indicate size of values in bytes. PVT values are in an Earth centered, Earth fixed (ECEF) coordinate system.

Table 2 PPS time message definition

PPS Time Packet MID: 0xB1

Offset	Format	Units	Description
0	U4	ms	GPS millisecond time of week
4	I4	ns	Fractional nanosecond of above ms
8	I2	n/a	GPS week
10	U1	n/a	Fix Type: 0 = NoFix 1 = Dead reckoning 2 = 2D fix 3 = 3D fix 4 = GPS + dead reckoning 5 = Time only fix
11	U1	n/a	Receiver Status, bits 0 – bits 3 Bit 0 : 1 if GPS fix ok Bit 1 : 1 if differential solution used Bit 2 : 1 if week number valid Bit 3 : 1 if time of week valid
12	U4	n/a	Padding, 0x00000000

Table 3 Position message definition

Position Packet MID: 0xB2

Offset	Format	Units	Description
0	I4	cm	ECEF X coordinate
4	I4	cm	ECEF Y coordinate
8	I4	cm	ECEF Z coordinate
12	U4	cm	3D position accuracy

Table 4 Velocity message definition

Velocity Packet MID: 0xB3

Offset	Format	Units	Description
0	I4	cm/s	ECEF X velocity
4	I4	cm/s	ECEF Y velocity
8	I4	cm/s	ECEF Z velocity
12	U4	cm/s	3D velocity accuracy

Table 5 No solution message definition

No Solution Packet MID: 0xB0

Offset	Format	Units	Description
0	U16	n/a	Empty packet all bytes 0x00

3. MATLAB Parsing Software

The reference MATLAB software for parsing the SPVTCG data is included in Appendix B. The software is organized into 3 functions. The top-level function is "ParseGPSTelStream," which takes 2 raw binary vectors containing the SPVTCG clock and data channels as extracted from the file produced by the specific data collection system used and the sample rate used during data collection. The sample rate is only used to estimate the blanking windows for frame alignment between PPS events and has no impact on timing accuracy. This function outputs 3 matrices containing PPS records, position records, and velocity records indexed to sample count. For example, the PPS records state at sample X what the GPS week and GPS ms into that week were at that sample point. This function locates the data packet frames and then breaks the bit stream into 20-byte message packets. The second function, "ParseGPSTelPAcket," is then called to extract information from individual packets. During this function call, the final function, "CalcFletcherCHKSum," is called to valid to the packet to ensure that only valid results are returned.

4. Conclusions and Future Developments

The SPVTCG provides low bandwidth, precise reference channels for distributed multiple sensor data collection activities. The precise synchronization is required for both fusion of information and comparison to truth data during algorithm development. The SPVTCG goes beyond normal IRIG time code generation by including position and velocity tightly coupled to the recorded data. This allows the data to be easily segmented and retain full state integrity, as well as be collected from mobile units. At the time of this report, the SPVTCG was a first-generation device and used successfully by ARL in support of mission-related activities.

A second generation is being considered with enhancements. The new features would include differential output signals to support long transmission lengths in high electromagnetic interference (EMI) environments. It would improve compatibility with audio voice recorders with strong DC blocks by implementing non-return to zero encoding schemes, such as Manchester encoding on the data line. Adding local PVT logging would allow a dual use role in field data collection when tracking target items. Size and power consumption would also be looked at in any future development.

Appendix A. GPS Time Code Generator Schematics

Figures A-1 and A-2 provide the GPS time code generator schematics.

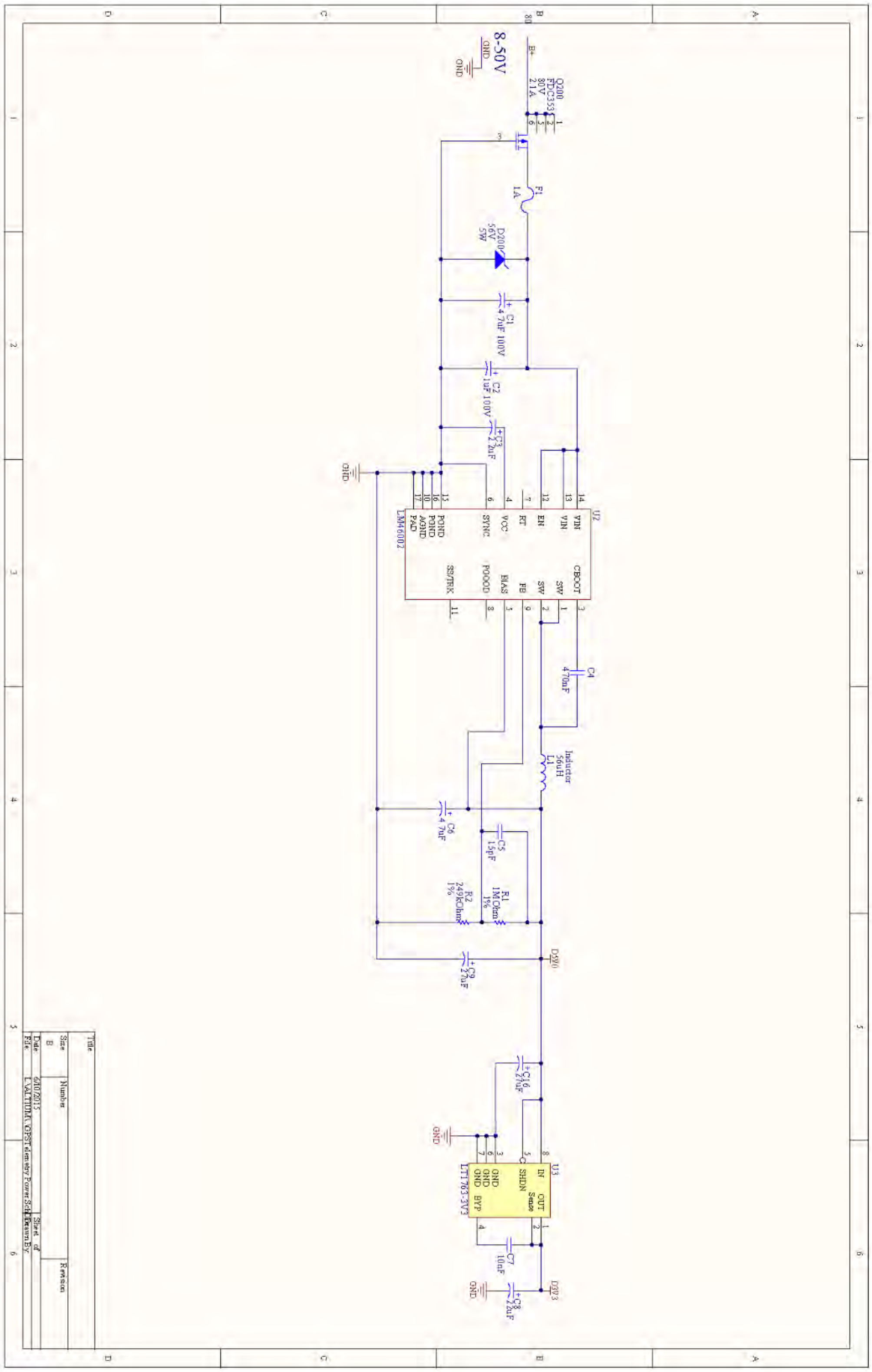

Fig. A-1 Power supply schematic

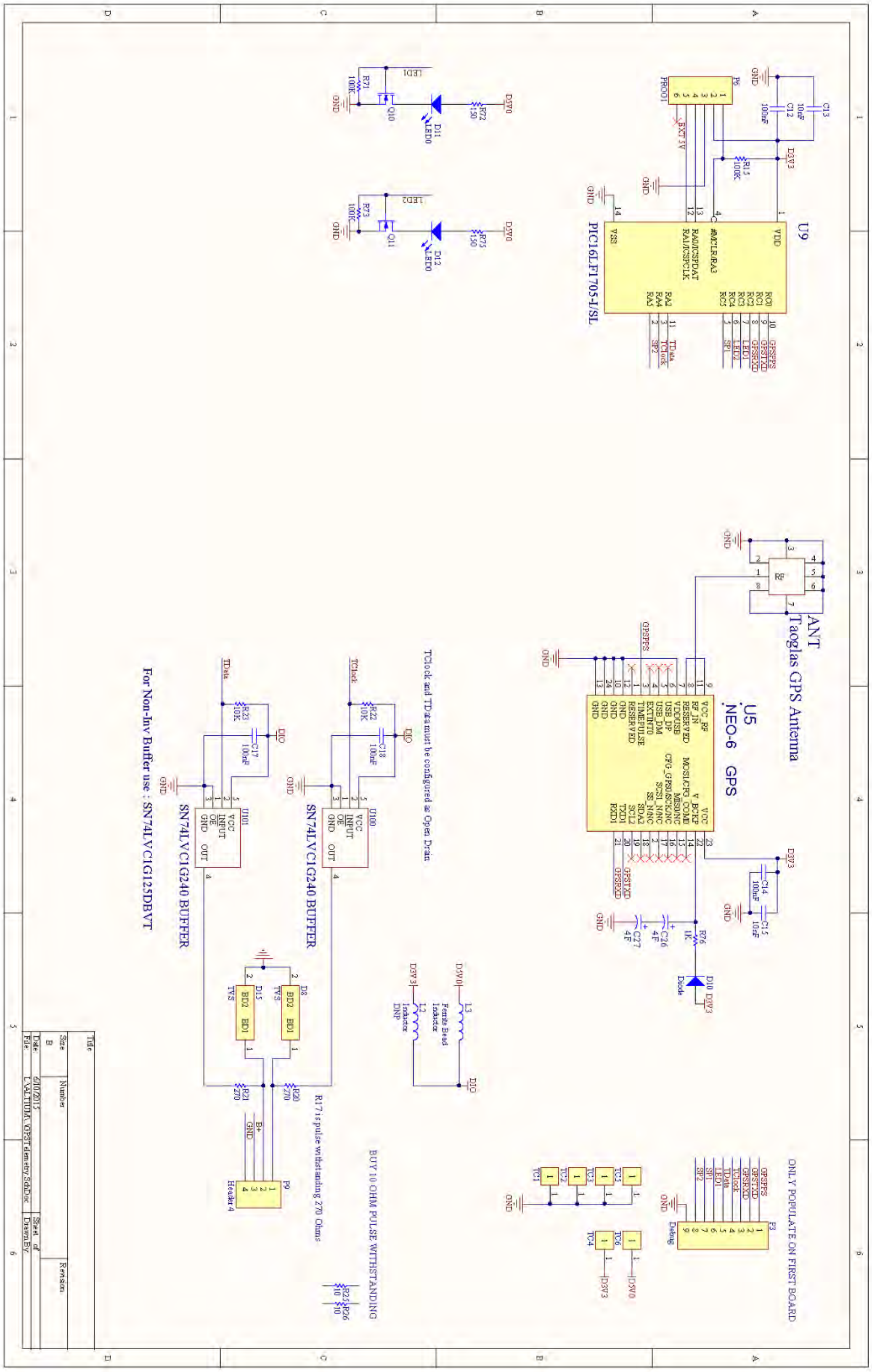

Fig. A-2 Digital logic schematic

INTENTIONALLY LEFT BLANK.

Appendix B. MATLAB Source Code

The following is the MATLAB source code:

```
function [ PPSRec, POSRec, VELRec] = ParseGPSTelStream( clk,data,sampleRate,debug
)
%ParseGPSTelStream( clk,data,sampleRate,debug )
%Extracts the Time Position and velocity reports from a SPVTCG bit streams.
%   Both clk and data are assumed equal length vectors with binary values
%   of 0 or 1
%   sampleRate is the digitization rate of the PVTCG stream and is only
%   used to approximate the blanking interval.
%   debug - If defined will plot detected PPS start event times for visual
%   confirmation
%
%   Outputs:  All outputs are indexed relative to input sample index,
%             All Locations and velocities in Earth Centered Earth Fixed
%       PPSRec: [ Event index , GPS Week, GPS millisec into week]
%       POSRec: [ Event index , PosX (meters),PosY (m),PosZ (m) ]
%       VELRec: [ Event index , VelX (m/s),VelY (m/s),VelZ (m/s) ]
%
%   BTM 6/2015 Rev 1.0
%

% test for debug flag
if nargin == 3
    debug = 0;
end

 nums = ceil(max(size(clk))/sampleRate);
%Pre-allocate storage to speed time, trim at end once discovery complete
%PPSRec contains index, GPS week, and ms into week
PPSRec = zeros(3,nums+1)';
NumPPSRec = 0;
%POSRec contains index, ECEF-X,ECEF-Y,ECEF-Z,Pos Accuracy
POSRec = zeros(5,nums+1)';
NumPOSRec = 0;
%VelRec contains index, Vx,Vy,Vz,Vel Accuaracy
VELRec = zeros(5,nums+1)';
NumVELRec = 0;


%The data rate is 200 Baud and the data clock is 400 Hz
%After each PPS event 20 bytes are sent then a 200 ms blank window in
%data and clock. This allows easy framing of the data
BlankWindowSamples = 0.2*sampleRate;
DataWindowSamples = 0.8*sampleRate;

%data is valid on faling endeges of clock extract data bits
fallingEdges = find(diff(clk) == -1) + 1;
dataFrameIndex = fallingEdges(find(diff(fallingEdges) > 0.1 * sampleRate) + 1);
dataBits = data(fallingEdges);

%find rising edges to locate PPS, after large 100 ms gaps
risingEdges = find(diff(clk) == 1) + 1;
ppsIndexs =  risingEdges(find(diff(risingEdges) > 0.1 * sampleRate) + 1);

if debug == 1
    figure
    plot(clk)
    hold
    plot(ppsIndexs,ones(max(size(ppsIndexs))),'+R');
end

%For each pps event extract that data message and parse, 160 data bits

for frame =  1:max(size(dataFrameIndex))
    %get index list for 120 bits
    firstBitIndex = find(fallingEdges == dataFrameIndex(frame));
    %handle last frame may be partial, test end index
    if (firstBitIndex+159) < max(size(dataBits))
        binaryFrame = dataBits(firstBitIndex:firstBitIndex+159);
```

```
        %Convert binary frame into byte packet, note data is LSB first
        bytes = reshape(binaryFrame,8,20)';
        b2uint8=[1 2 4 8 16 32 64 128]';
        packet = uint8(bytes * b2uint8);
        if debug
            dec2hex(packet)'
        end
        Results = ParseGPSTelPacket( packet );
        switch Results.type
            case 'TIME'
                NumPPSRec = NumPPSRec + 1;
                PPSRec(NumPPSRec,:) = [ppsIndexs(frame), Results.GPSWeek,
Results.miliSecWeek];
            case 'POS'
                NumPOSRec = NumPOSRec + 1;
                POSRec(NumPOSRec,:) = [ppsIndexs(frame), Results.POSX,
Results.POSY, Results.POSZ, Results.PACC];
            case 'VEL'
                NumVELRec = NumVELRec + 1;
                VELRec(NumVELRec,:) = [ppsIndexs(frame), Results.VELX,
Results.VELY, Results.VELZ, Results.VACC];
        end

    end

end


%Done so trim outputs to final length
PPSRec = PPSRec(1:NumPPSRec,:);
POSRec = POSRec(1:NumPOSRec,:);
VELRec = VELRec(1:NumVELRec,:);
end



function [ Results ] = ParseGPSTelPacket( packet )
%PARSEGPSTELPACKET (packet) Extract information fron SPVTCG Packet
%   Input is packet vector of 20 bytes(uint8)
%
%   Output: Parsed and converter results structure with type field
%
%
%   BTM 6/2015 Rev 1.0
%

  % Confirm starts as SPVTCG packet and the packet valid
  if(packet(1) == hex2dec('A5') && (CalcFletcherCHKSum(packet) == 1))
      switch packet(2)

          case hex2dec('B0')
            Results.type = 'NOSOL';

          case hex2dec('B1')
            Results.type = 'TIME';
            Results.miliSecWeek = double(typecast(packet(3:6),'uint32'));
            Results.fracNano = double(typecast(packet(7:10),'int32'));
            Results.GPSWeek = double(typecast(packet(11:12),'int16'));
            Results.fix = typecast(packet(13),'uint8');
            Results.flags = typecast(packet(14),'uint8');

          case hex2dec('B2')
            Results.type = 'POS';
            Results.POSX = double(typecast(packet(3:6),'int32'))/100;
            Results.POSY = double(typecast(packet(7:10),'int32'))/100;
            Results.POSZ = double(typecast(packet(11:14),'int32'))/100;
            Results.PACC = double(typecast(packet(15:18),'uint32'))/100;

          case hex2dec('B3')
            Results.type = 'VEL';
            Results.VELX = double(typecast(packet(3:6),'int32'))/100;
```

```
            Results.VELY = double(typecast(packet(7:10),'int32'))/100;
            Results.VELZ = double(typecast(packet(11:14),'int32'))/100;
            Results.VACC = double(typecast(packet(15:18),'uint32'))/100;
      end

  else
      Results.type = 'NONE';
  end
end


function [Valid] = CalcFletcherCHKSum(InputPacket)
%CalcFlectcherSHKSum(InputPacket)
%   This function validates the 8-bit Flecher Checksum Algorithm defined in
RFC(1145)
%   InputPacket is a vector of bytes (uint8) with the input checksum in last 2
bytes
%
% Function returns 1 (valid packet) if calculated checksum = input checksum
%
% The checksum calculation specifies the 2 checksum values to be 8 bit
% integers and the the calculations rollover at 256. Matlab saturates when
% calculating the uint8 math at 255. For this reason the checksum
% intermediates are uint32's then only the final low 8 bits are used as the
% final checksum value via the mod(256).
%
%   BTM 6/2015 Rev 1.0
%

    CK_A = uint32(0);
    CK_B = uint32(0);
    data = uint32(InputPacket);
    len = max(size(data));
    InCK_A = data(len - 1);
    InCK_B = data(len);
    for i = 1:(len - 2)
        CK_A = mod(CK_A + data(i),256);
        CK_B = mod(CK_B + CK_A,256);
    end
    res1  = CK_A;
    res2  = CK_B;
    if (res1 == InCK_A) && (res2 == InCK_B)
        Valid = 1;
    else
        Valid = 0;
    end
end
```

List of Symbols, Abbreviations, and Acronyms

ARL	US Army Research Laboratory
ECEF	Earth centered, Earth fixed
EMF	electromagnetic interference
GPS	global positioning system
IRIG	Inter-Range Instrumentation Group
ISR	intelligence surveillance and reconnaissance
MID	message identification
NMEA	National Marine Electronics Association
PCB	printed circuit board
PPS	pulse per second
PVT	position, velocity, and time
SOP	start of packet
SPVTCG	Synchronized Position, Velocity, and Time Code Generator
TTL	transistor-transistor-logic

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
18 (PDF)	US ARMY RSRCH LAB RDRL SES A B LISS J HOUSER N SROUR R DAMARLA T WALKER C REIFF RDRL SES J EICKE RDRL SES P H VU D HULL M SCANLON S TENNEY RDRL SES S A LADAS B MAYS M WARE B MARY D WARD M YOUNG RDRL SES X J HOPKINS